Lab 2

Object Recognition and 3D Pose Estimation

Objective

In this experiment, Students will continue using stereo cameras, an image acquisition program and image processing algorithms to identify different objects and estimate their positions (XYZ coordinates) as well as orientations (OAT angles) in the camera field of view. Once the poses of the objects are calculated, you are to move the robot to simulate the grasping for each object in the robot workspace. You are required to achieve the following goals.

1. Learn how to use image processing algorithms to identify different objects in an image.
2. Learn how to incorporate the camera calibration matrix obtained in the previous lab experiment to calculate the pose (position + orientation) of the objects in the robot workspace.
3. Learn how to manipulate PUMA 260 robot.
4. Define a successful grasping strategy for picking up different objects.

Reference Materials

1. Lecture notes, i.e. 3D reconstruction and image processing.
2. Matlab Image Processing Toolbox. Some of the functions that are very useful in this lab are: hough, houghlines, houghpeaks, edge, bwlabel, graythresh, bwconncomp, regionprops, etc.

Pre-lab

1. a) Please read the following materials: Lecture notes about "Stereo Vision", "3D Reconstruction", "Edge Detection", "Hough Transforms".
2. b) Explain the performance of the following MATLAB functions hough, houghlines, houghpeaks, edge, bwlabel, graythresh, bwconncomp, regionprops.

Lab Procedure

This lab consists of the following five parts:

Part 1: Image Processing

1. Take a snapshot of the robot workspace where four cylinders are in different poses. You can put white paper on the background in order to get strong contrast between the objects and the background.
2. Design a sequence of image processing algorithm such as, binarization (thresholding), component labeling, edge detection, Hough transform, etc. (Hint: refer to the examples under: /usr/local/src/LinuxPUMAs/src/Hough_Examples/)
3. At this point you should have obtained the centroid coordinate (u, v) and orientation for each object.

Part 2: 3D Reconstruction

In this case you will use the Calibration toolobox.

1. Once the (u, v) centroid coordinates of the objects are calculated, you can use the camera calibration matrix to reconstruct the XYZ coordinate in the camera coordinate. You may need to re-calibrate the camera using the calibration toolbox in case someone touched the camera by accident.
2. The objects to be used are black cylinders with different lengths and diameters and the algorithm should be able to output a list of all objects with their XYZ positions and orientations on the XY plane.

Part 3: Start the Puma 260 Robot

1. Login to one of the computers in the lab.
2. Start minicom from the command window, by simply typing:
   $ minicom –b 9600 –D /dev/ttyUSB0.
3. Turn on the AC power of the robot controller (The upper key). A message will appear on minicom saying:
   .Initialize(Y/N)?
   Usually we select Y to clear all the previous data.
4. Turn on the ARM power (lower key).
5. On the minicom window, type :
   .calibrate
   A question "Are you sure" will appear. Usually select Y.
6. Type :
   .do ready

"ready" is a predefined location and the robot must be moved to this position both after automatic calibration and before the robot nests.

7. Type:
   .speed 20
   Usually we set the speed 20 or 10 for the safety consideration
8. Type :
   .do nest
   This step is carried out in order to check the calibration process do ready takes the robot to the ready position after the calibration process is done.
9. Type:
   .here Object
   You are to define the XYZOAT coordinate of the detected object, where the XYZ are from the 3D reconstruction in Part 2. OAT angles are related to the pose to pick the object, which are described in Part 4.
   Note: You should have a clear understanding of the meaning of OAT angles. You might check the VAL II user manual for their definitions.
10. Type:
   .do move Object
   The robot will move to the destination you just defined in step 9

Part 4: Control the Robot with C++ interface

1. Besides the minicom interface, an alternative interface written with C++ is provided to manipulate the robot. You will find the source codes under (/usr/local/src/LinuxPUMAs/src/Puma_API) as well as the ready binaries under (/usr/local/bin)
2. The incorporation of C/C++ binaries with matlab code requires system function call. Please refer to the sample code in (/usr/local/src/LinuxPUMAs/src/PumaMove) for more details.

Part 5: Picking Objects

1. Based on the pose (position + orientation) of the object, define a grasping strategy for the PUMA 260. The gripper is mounted to the end-effector. That implies that only a certain orientation of the gripper can be used to grasp the object. For example, you can choose to align the Y axis of the end-effector with the orientation of the cylinder when it is flat on the table. (Hint: remember to take into account the length of the gripper when setting the Z

coordinate)

NOTE: To open the gripper, you can run `openGripper` and to close the gripper you can run `closeGripper`. You can find source codes under `/usr/local/src/LinuxPUMAs/src/gripper`.

2. Move cylinders closer to each other, could your algorithm still be able to correctly identify the number of objects? If not, why? Do you have a better way to guarantee the success in the image processing part? What if there are some shades in the environment (in case of there is a box containing the cylinders)? Could your algorithm distinguish between the objects and the shades?

Post-Lab Questions:

Comment your results by answering to below questions:

- How you handle detection of objects with different size, close to each other, with shadow?
- What did you use for OAT angles? Were there any issues? How did you solve?